Page tables

XV6 Runs on SV39 RISC.

PT → 4096 byte 2^{12}.

RISV page table is logically an array of 2^{27}. PTE.

Context Switch

↓

happens during I/O or Interrupt

If Timer interrupt, give up CPU.
if (dev==2) – yield
it is timer interrupt.

yield → proc.c
↓
takes current process state
saves it & resumes it ~~later~~.

save all registers callee register.

ra → return address

sp → stack point

process is running, CPU wants to switch to another process2 then....
→ there will be a timer interrupt
→ call yield ( ) in trap.c
↳ turns a RUNNING process to

yie
↓
yield
double ↓
calls

context · sw
shed() ↓
↳ calls
it is
ld

it doent k
process is calle
it just s
one &
and load

RUNNABLE
yield is in proc.
↓
yield calls sched
↓
double check all conditions
↓
calls switch to switch the context.

sched()
↳ calls switch (sp → content,)
↓
it is an assembly code
ld , sd

it doest know anything about which process is called etc
it just takes a new context & an old one & save old context registers and load from new.

```
sd , ra , 0(a0)
sd , sp , 8(a0)
;

ld  ra  0(a1)
ld ,sp, 8(a1)
```

now cpu will start at address indicated at new ra

- Scheduler → proc.c

each CPU calls scheduler after setting them up, It is never ending.

→ keep interrupt ON

```
for (p=proc ; p < &proc [NPROC] ; p++){
    acquire ( &p → lock )
```

Uthreads.c

in thread-init (which is called first) we initialise all-thread[0] with RUNNING ~~RUNNABLE~~ state & current-thread= &all-threads[0],

~~than~~ If then control goes to create-thread. then schedule. "we make threads RUNNABLE

In schedule

```
1st Iter    t = current-thread + 1        &all-thread[1]

            for ( i < MAX-THREADS)
               if ( t >= all-threads + MAX-THREAD)
                     t = all-thread
               if ( t → state = RUNNABLE )
                  {
                       next-thread = t;  // here we get our next Runnable thread
                  }  break

            t=t+1;
```

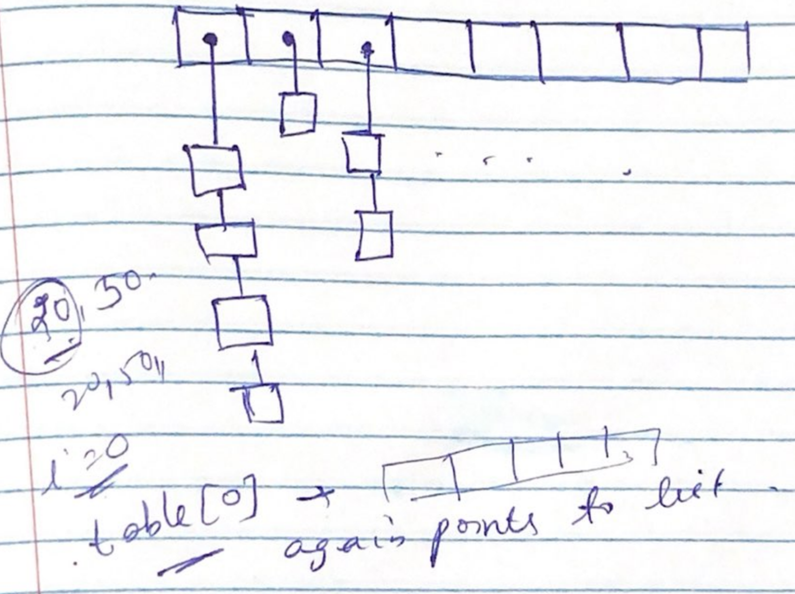
② struct entry *table [NBUCKET]

array of pointers that point to ~~structures [pointers to array]~~.

```
struct entry {
    int key;
    int value;
    struct entry *next ;
  };
```

now entry has a pointa next, so.

*table [NBUCKET] , each ~~array~~ element can point to either one struct or even a list.head.


20, 50.
20, 50
i=20
table[0] →
again points to list